UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/006,551	11/30/2001	Christopher D.S. Donham	NVIDP064/P000286	2643

28875 7590 09/18/2009
Zilka-Kotab, PC
P.O. BOX 721120
SAN JOSE, CA 95172-1120

EXAMINER	
AMIN, JWALANT B	

ART UNIT	PAPER NUMBER
2628	

NOTIFICATION DATE	DELIVERY MODE
09/18/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

anita@zilkakotab.com
erica@zilkakotab.com
dottie@zilkakotab.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte CHRISTOPHER D.S. DONHAM, EDWARD HUTCHINS,
ALEXANDER MINKIN, and GEORGE E. SCOTT, III

Appeal 2009-003210[1]
Application 10/006,551
Technology Center 2600

Decided: September 16, 2009

Before JOHN C. MARTIN, JOSEPH F. RUGGIERO,
and ROBERT E. NAPPI, *Administrative Patent Judges.*

MARTIN, *Administrative Patent Judge.*

DECISION ON APPEAL

[1] The real party in interest is NVIDIA Corporation. Br. 3.

STATEMENT OF THE CASE

This is an appeal under 35 U.S.C. § 134(a) from the Examiner's rejection of claims 1-30, which are all of the pending claims.

We have jurisdiction under 35 U.S.C. § 6(b). We affirm.

A. Appellants' invention

Appellants' invention relates to texture mapping in a computer graphics processing pipeline. Specification 1:7-8. As explained in more detail below, the invention differs from the prior art described in the Application by using a texture module in a graphics pipeline to retrieve "instructions" from a video memory.

Appellants' Figure 2, which is labeled "Prior Art," is reproduced below.

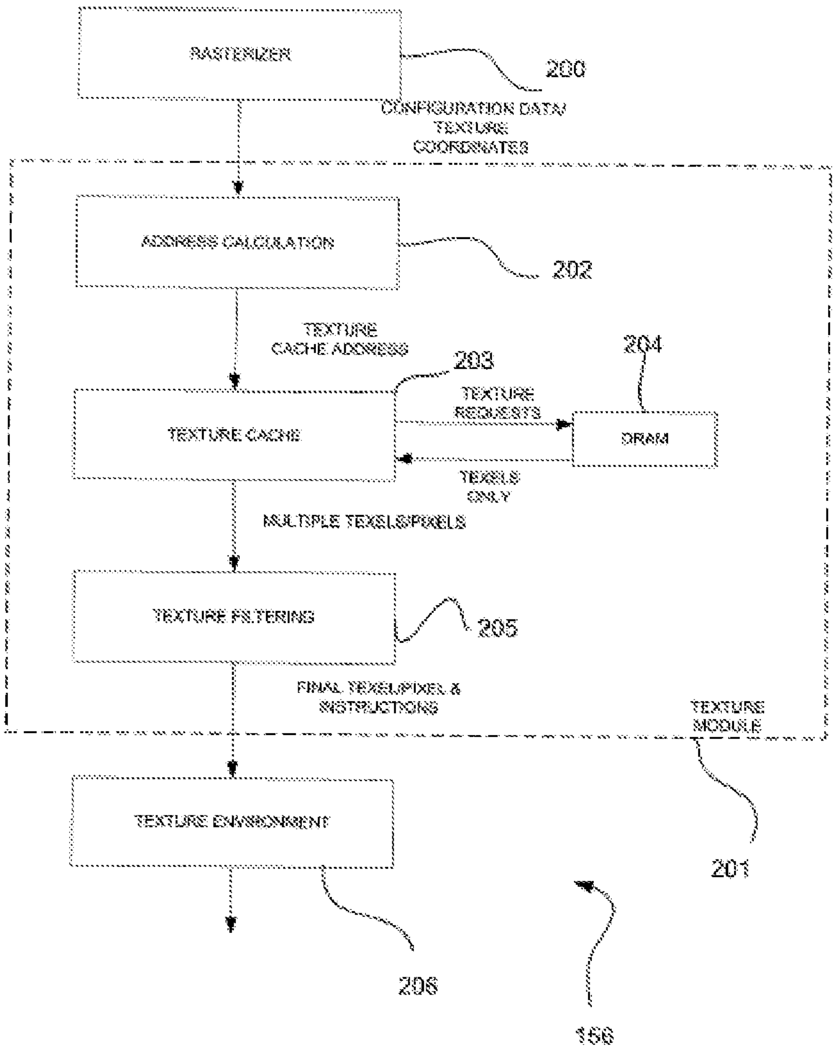

Figure 2
(Prior Art)

Prior Art Figure 2 is a more detailed diagram of the rasterization/ texture module 156 of Prior Art Figure 1 (not reproduced herein), which illustrates a graphics pipeline with which texture mapping may be performed. *Id.* at 8:8-12. As explained below, DRAM 204 stores textures in the form of texels (texture elements).

Rasterizer 200 passes configuration data (i.e., instructions[2]) for control registers to the texture module 201 and to the texture environment module 206. *Id.* at 2:22-27. In addition, rasterizer 200 generates texture coordinates for all of the pixels that comprise a primitive. *Id.* at 2:27-28.[3] In response to the receipt of such texture coordinates, texture module 201 is adapted to calculate texture cache addresses utilizing an address calculation module 202. *Id.* at 2:28-30. The texture cache addresses may be then utilized for looking up corresponding textures in the form of texels from memory 204 utilizing a texture cache 203. *Id.* at 2:30 to 3:2. Upon receipt of a texture cache address, texture cache 203 outputs the associated texels to a texture filtering module 205 if the texels are already present inside texture cache 203. *Id.* at 3:2-4. If the associated texels are not present in texture cache 203, texture requests are sent to memory 204. *Id.* at 3:4-6. In response to texture requests, the memory 204 sends the associated texels to texture cache 203. *Id.* at 3:6-7.

The filtered texels from texture filtering unit 205 are blended by texture environment module 206 with other pixel data (such as other,

[2] In describing their invention, Appellants explain that the instructions used therein may take the form of "control information, configuration data, etc." for the graphics pipeline. *Id.* at 10:6-7.

previously referenced textures). *Id.* at 3:15-17. Once all texture blending operations are completed, the pixel data is passed on to the frame buffer (not shown). *Id.* at 3:17-18.

In recent prior art, the texture environment 206 has been replaced with pixel shading hardware, such as shown in Figure 3, reproduced below.

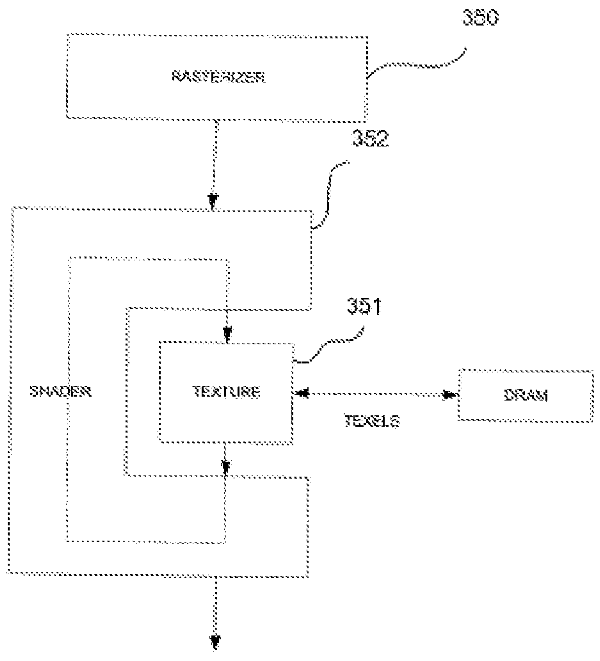

Figure 3
(Prior Art)

Figure 3, another detailed diagram of the rasterization/texture module 156 of Prior Art Figure 1 (*id.* at 8:11-12), illustrates an example of a modern, high performance system. *Id.* at 4:25-26. While rasterizer 200 and texture

[3] A "graphic primitive" is a basic component of a graphic picture, such as a polygon, a triangle, a line or a point. *Id.* at 1:15-17. All graphic pictures are
(Continued on next page.)

module 201 still exist in the high performance system (see rasterizer 350 and texture module 351), texture environment module 206 has been replaced with a shader module 352. *Id.* at 4:26-28. One of the key advantages of this architecture is that processing for a primitive can "loop" (i.e., the texels resulting from one texture lookup can influence the location of the texels in a subsequent texture lookup). *Id.* at 4:28-31. The configuration data that controls this looping and its associated math operations is typically referred to as a "shader program" because it bears a resemblance to a "program" that a general purpose computer would execute. *Id.* at 5:4-6.

With the expanding options enabled by the architecture in Prior Art Figure 3, there is an associated increase in the amount of configuration data required to achieve a desired effect. *Id.* at 5:8-10. Thus, there is a need to accommodate the programmability of recent texture and shader modules without being inhibited by the size of associated programs. *Id.* at 5:13-15.

Appellants' solution is depicted in Figure 3A, reproduced below.

---

formed with combinations of these graphic primitives. *Id.* at 1:17-18.

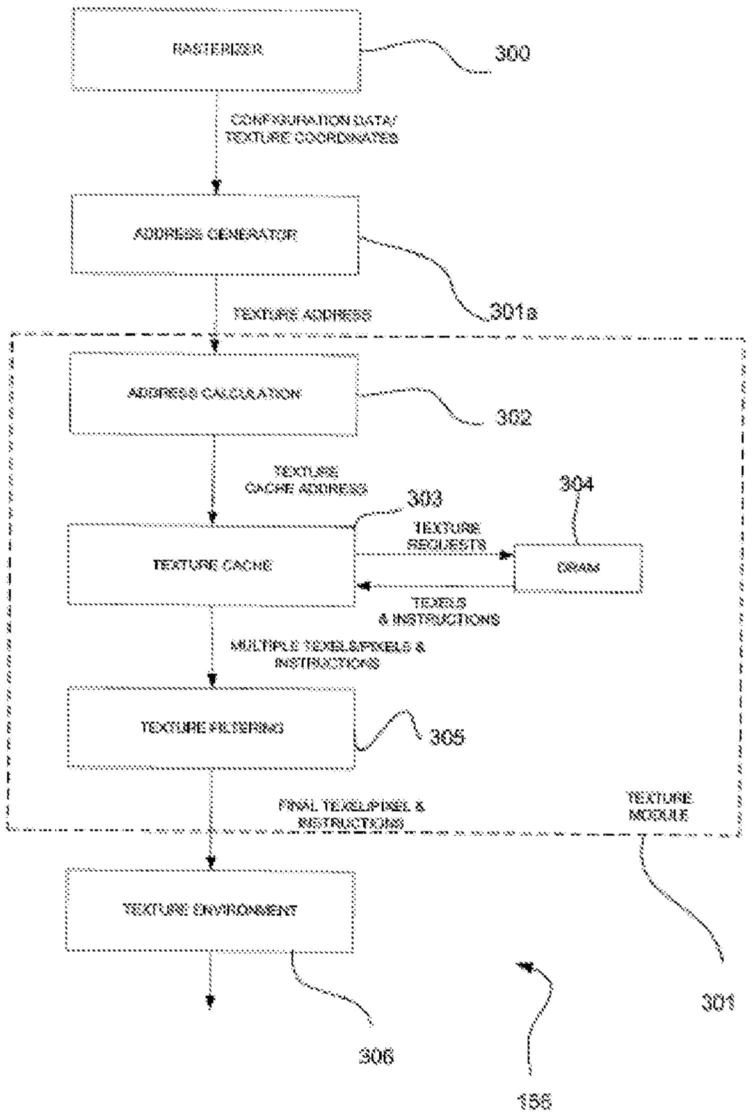

Figure 3A

Figure 3A illustrates an embodiment of a rasterization/texture module in accordance with Appellants' invention that is capable of looking up instructions. *Id.* at 8:14-15.

An address generator 301a is provided for converting instruction requests from the rasterizer 300 into pseudo-texture coordinates for the texture module 301. *Id.* at 9:19-20. Address calculation module 302 computes texture cache addresses for the pseudo-texture coordinates based on mode and state information. *Id.* at 9:23-25. Texture cache 303 returns the pseudo-texel data if the data for the texture cache address is in cache memory. *Id.* at 9:26-27. Otherwise, texture cache 303 requests the associated data from memory 304 and, upon receipt of the pseudo-texel data from the memory 304, stores this data in local memory, and outputs this data to a texture filter module 305. *Id.* at 9:27-30. The texture filtering module 305 processes pseudo-texel data in accordance with state and mode information and outputs the pseudo-texel data to a texture environment module 306. *Id.* at 9:30 to 10:2.

The pseudo-texel data is not necessarily mapped onto a primitive and does not necessarily represent conventional visual data (i.e. color data). *Id.* at 10:4-6. Instead, the pseudo-texel data may represent instructions (i.e. control information, configuration data, etc.) for the graphics pipeline. *Id.* at 10:6-7. By retrieving the instructions from memory 304 utilizing texture module 301, much pipeline bandwidth is saved at the input of texture module 301 since the prior art configuration data need not be received from rasterizer 300. *Id.* at 10:29 to 11:1. The instructions may then be used by texture module 301 and subsequent modules in order to control various graphics processing involving the texels, pixels, and/or primitives, etc. *Id.* at 11:3-5.

Although Figure 3A depicts DRAM 304 as part of texture module 301, the Specification suggests that it can be separate therefrom by stating that "the memory 304 traditionally employs a high-bandwidth connection with the texture module 301." Specification 11:1-2.

B. The claims

The independent claims before us are claims 1 and 24-30, of which claim 1 reads as follows:

> 1. A method for execution with a system including a tangible computer readable medium, the method for retrieving instructions from video memory utilizing a texture module in a graphics pipeline, comprising:
>
> (a) sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory; and
>
> (b) receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline.

Claims App., Br. 79.

Claim 1, in reciting that the "texture module . . . sends the instruction request to the video memory," implicitly requires that the texture module be separate from the video memory.

C. The references and rejection

The Examiner relies on the following prior art:

Wang et al. (Wang)	US 5,831,640	Nov. 3, 1998

Rivard et al. (Rivard)	US 5,987,567	Nov. 16. 1999

"Applicant Admitted Prior Art," i.e., the prior art system depicted in Appellants' Prior Art Figure 3 (Specification 19, para. 38), which employs a texture module and a shader module (hereinafter "AAPA").

Rejection 1: Claims 1-12, 18-21, 24-28, and 30 stand rejected under 35 U.S.C. § 103(a) for obviousness over Rivard in view of Wang. Final Action 13.

Rejection 2: Claims 13-17, 22, 23, and 29 stand rejected under § 103(a) for obviousness over Rivard in view of Wang and AAPA.

Appellants have divided the claims, for argument purposes, into the following issues and groups:

Issue 1 (i.e., Rejection 1)

Group 1: Claims 1-12, 21, 24, and 27 (Br. 12), of which we select claim 1 as a representative claim pursuant to 37 C.F.R. § 41.37(c)(1)(vii);

Group 2: Claims 25 and 26 (*id.* at 30);

Group 3: Claim 28 (*id.* at 41);

Group 4: Claim 30 (*id.* at 50);

Group 5: Claim 18 (*id.* at 62); and

Group 6: Claims 19 and 20 (*id.* at 64).

Issue 2 (i.e., Rejection 2)

Group 1: Claims 13-17, 22, and 23 (*id.* at 67); and

Group 2: Claim 29 (*id.*).

THE ISSUES

Appellants have the burden on appeal to show reversible error by the Examiner in maintaining the rejections. *See In re Kahn*, 441 F.3d 977, 985-86 (Fed. Cir. 2006) ("On appeal to the Board, an applicant can overcome a rejection by showing insufficient evidence of *prima facie* obviousness or by rebutting the *prima facie* case with evidence of secondary indicia of nonobviousness." (citation omitted)).

The principle issue regarding the rejection of representative claim 1 is whether Appellants have demonstrated that the Examiner erred in concluding it would have been obvious in view of Wang to store instructions as well as texture data in Rivard's DRAM 655.

THE SCOPE AND MEANING OF "INSTRUCTIONS"

The claims must be interpreted as broadly as is reasonable and consistent with the specification, *In re Thrift*, 298 F.3d 1357, 1364 (Fed. Cir. 2002), while "taking into account whatever enlightenment by way of definitions or otherwise that may be afforded by the written description contained in the applicant's specification," *In re Morris*, 127 F.3d 1048, 1054 (Fed. Cir. 1997), and without reading limitations from examples given in the specification into the claims, *In re Zletz*, 893 F.2d 319, 321-22 (Fed. Cir. 1989).

The Specification does not define any of the claim terms instructions," "video memory," and "texture module."

The Examiner (Final Action 3) relies on definition 6 of the definitions for "instruction" given at dictionary.com:

> 1. the act or practice of instructing or teaching; education.
> 2. knowledge or information imparted.
> 3. an item of such knowledge or information.
> 4. Usually, instructions. orders or directions: *The instructions are on the back of the box.*
> 5. the act of furnishing with authoritative directions.
> 6. *Computers.* a command given to a computer to carry out a particular operation.

http://dictionary.reference.com/browse/instruction (last accessed Aug. 25, 2009). In addition to quoting definition 6, the Examiner added that an instruction "can contain data to be used in the operation," which appears to be directed to the fact that Wang (discussed *infra*) discloses "display instructions includ[ing] texture data." Wang, col. 5, l. 47. Appellants have not offered any other definition of "instruction" and do not deny that definition 6 is an appropriate definition. *See* Reply Br. 40 ("[T]the dictionary.com reference provided by the Examiner merely defines an instruction as 'a command given to a computer to carry out a particular operation.'").

In view of the foregoing definition and the fact that Appellants' "instructions" can take the form of "control information, configuration data, etc." for the graphics pipeline (*id.* at 10:6-7), we understand the claim term "instructions" to be broad enough to read on any type of information that is described as being used to control any aspect of the operation of the graphics pipeline, such as the manner in which texture data is processed. Thus, we do not understand the term “instructions” to be readable on the texture data that is being processed by the graphics pipeline.

RIVARD

Rivard discloses a graphics rendering system that employs a graphics accelerator system for caching the most-recently-read texels and the adjacent raster line of texels for use in interpolative sampling to compute dynamic texture values. Rivard, col. 2, l. 66 to col. 3, l. 5.

Rivard’s Figure 6 is reproduced below.

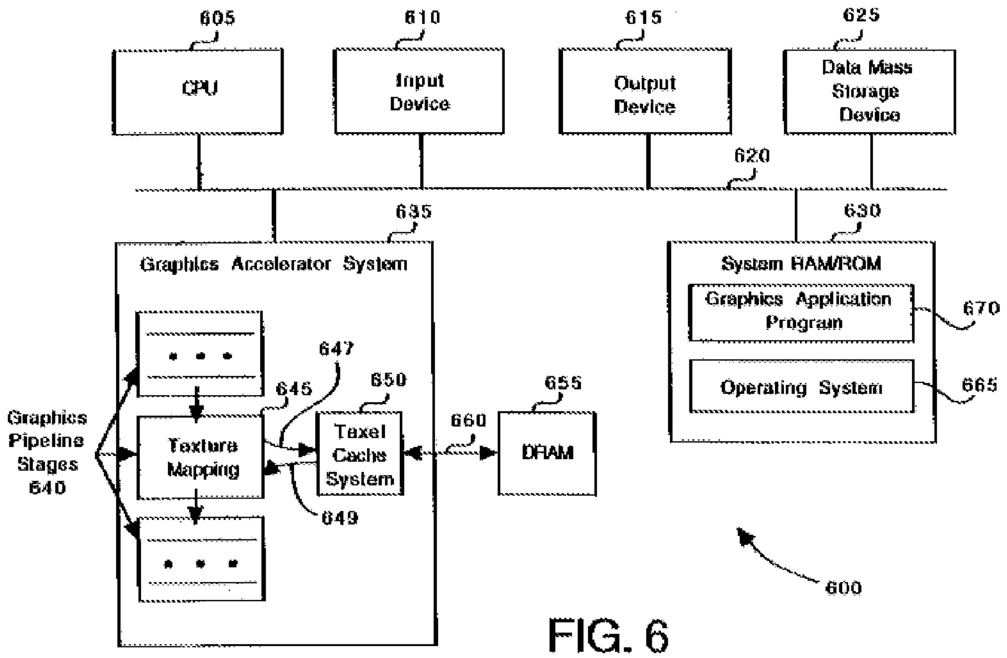

FIG. 6

Figure 6 is a block diagram illustrating Rivard’s computer system. *Id.* at col. 3, ll. 54-55.

Appellants do not question the Examiner’s finding that the claim term “video memory” reads on Rivard’s DRAM 655. Final Action 13.

The Examiner reads the recited "texture module" on the combination of texture mapping stage 645 and texel cache system 650. Specifically, the Examiner found that "Rivard does not explicitly teach to combine texture mapping stage [645] and texel cache system [650] to form a texture module" and concluded that "it would have been obvious to one of ordinary skill in art at the time of present invention to combine texture mapping stage and texel cache system of Rivard to work together as a texture module." *Id.* at 15. Appellants appear to be of the view that the claim term "texture module" is readable only on texture mapping stage 645 and thus is not readable on texture mapping stage 645 in combination with texel cache system 650. *See* Br. 17 ("Clearly, the texture mapping stage <u>is separate from</u> the texture cache system, as clearly disclosed in Rivard, and thus fails to suggest 'sending an <u>instruction request</u> to video memory, where a <u>texture module</u> . . . sends the instruction request to the video memory' (emphasis added), as claimed by appellant[s]"). We do not agree. Appellants have not demonstrated that the term "texture module" cannot be read on the combination of texture mapping stage 645 and texel cache system 650. Furthermore, even assuming that "texture module" is readable only on Rivard's texture mapping stage 645, the claim language does not preclude the "texture module" from sending an instruction to the "video memory" through an intermediate element, such as Rivard's texel cache system 650.

The Examiner further found that Rivard's texture mapping stage 645 and texel cache system 650 (collectively corresponding to the recited "texture module") send an "instruction request" to DRAM 655 (Final Action

4-5), as required by claim 1, but found that "Rivard does not explicitly teach that the memory returns instructions along with data, in response to instruction request from the texture module" (Answer 5), as also required by claim 1, for which teaching the Examiner relies on Wang. *Id.*

Because some of Appellants' arguments (discussed *infra*) are based on "pipeline latency elements" employed in Rivard's texel cache system 650, we will briefly discuss Figure 10, which is reproduced below.

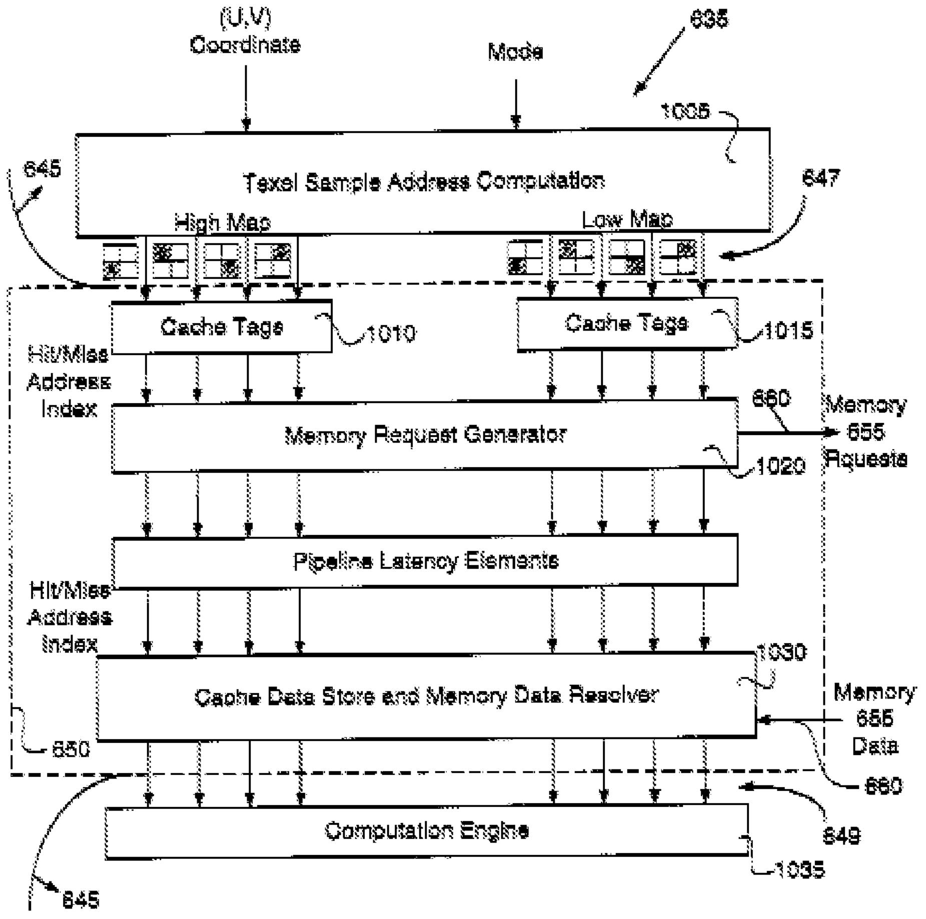

FIG. 10

Figure 10 is a block diagram detailing graphics accelerator system 635 and showing that texel cache system 650 includes cache tags 1010, 1015 a memory request generator 1020, pipeline latency elements 1025, and cache data store and memory data resolver 1030. *Id.* at col. 6, ll. 22-26.

Because memory request generator 1020 is located between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the associated address and instruction information reach cache data store and memory data resolver 1030. *Id.* at col. 6, ll. 62-66. Once memory requests are generated, there is, depending on the DRAM design, a constant latency of about five to ten clock cycles (or possibly more), which includes time for exiting and reentering the graphics pipeline hardware, to effect a page hit. *Id.* at col. 6, l. 6 to col. 7, l. 3. Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030. *Id.* at col. 7, ll. 3-7. Cache data store and memory data resolver 1030 receive the incoming memory data and incoming cache tag data, and store the values in cache memory. *Id.* at col. 7, ll. 8-10. Resolver 1030 formats and presents the data to a computation engine 1035, such as an Arithmetic Logic Unit (ALU). *Id.* at col. 7, ll. 10-12.

The Examiner, citing the above-discussed dictionary.com definition of "instruction" as "a command given to a computer to carry out a particular operation," found that the claim term "instruction request" reads on Rivard's "memory request" because DRAM 655 performs the operation of outputting memory data in response to the memory request. Final Action 3. Although Appellants take issue with this finding, it is not necessary to decide whether it is correct. The reason is that the question raised by the rejection is whether the claimed "texture module" (i.e., Rivard's texture mapping circuit 645 alone or in combination with texel cache system 650) will send an instruction request to DRAM 655 and receive an instruction from DRAM 655 in response to the instruction request when Rivard is modified in view of Wang. We therefore turn our attention to that reference.

Wang discloses a computer controlled graphics display system that includes a texture data cache controller unit that can perform useful texture data processing while waiting for fetched texture data associated with a texture cache miss. Wang, col. 2, ll. 28-34.

Figure 1 is reproduced below.

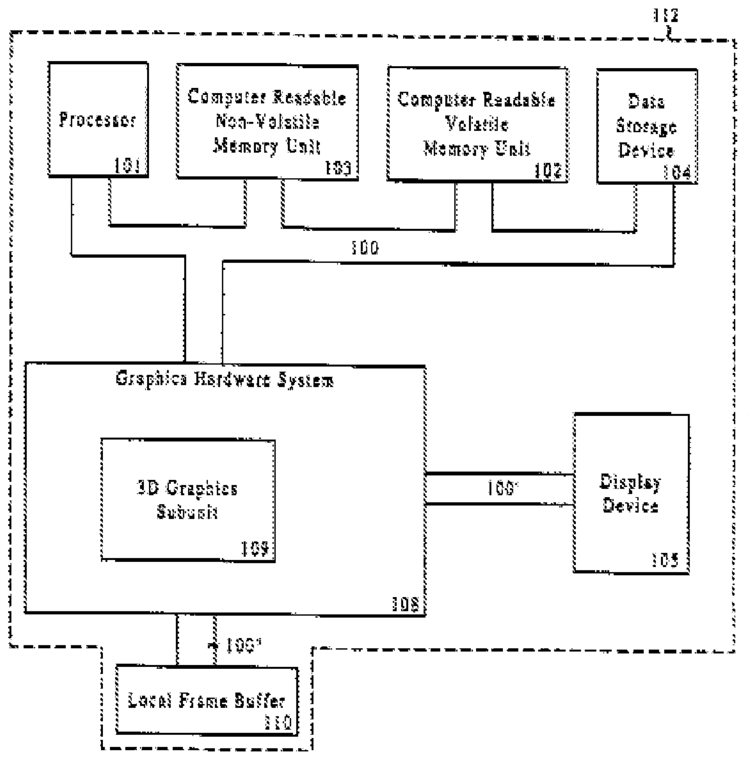

FIG. 1

Figure 1 is a block diagram of Wang's computer controlled graphics display system. *Id.* at col. 2, ll. 39-41.

Host computer system 112 comprises a bus 100 for communicating information, one or more host processors 101 for processing information and instructions, a computer readable volatile memory unit 102 (e.g. random access memory unit) for storing information and instructions for the host processor 101 or other PCI masters, a computer readable non-volatile memory unit 103 (e.g., read only memory unit) for storing static information and instructions for the host processor 101, a computer readable data storage device 104 such as a magnetic or optical disk and disk drive (e.g., hard drive or floppy diskette) for storing information and instructions, and a display device 105 for displaying information to the computer user. *Id.* at col. 5, ll. 17-33.

Host computer system 112 provides data and control signals via bus 100 to a graphics hardware unit or system (e.g., "graphics card") 108 that contains a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory. *Id.* at col. 5, ll. 38-43. The display list generally contains instructions regarding the rendering of several graphic primitives, e.g., individual points, lines, polygons, fills, BIT BLTs (bit block transfers), textures, etc. *Id.* at col. 5, ll. 43-46. Many of the polygon display instructions include texture data to be displayed within the polygon. *Id.* at col. 5, ll. 46-48. Texture data is stored in computer readable (e.g., volatile) memory units of system 112, or local frame buffer 110) in the form of raster based data (e.g., in one form its bit mapped) stored in (u,v) coordinates. *Id.* at col. 5, ll. 48-51. The individual components (e.g., "texels") of the texture data are read out of memory and applied within their polygon in particular fashions depending on the placement and perspective of their associated polygon. *Id.* at col. 5, ll. 51-55. The process of rendering a polygon with associated texture data, which is called "texture mapping," requires a great demand on the memory capacity of the computer system 112 because many texture maps are accessed from memory to construct a displayed frame. *Id.* at col. 5, ll. 55-60. Since screen updates need to be performed rapidly, polygons need to be updated very rapidly and further texture maps need to be accessed and applied in extremely rapid fashion, increasing memory demands. *Id.* at col. 5, ll. 60-63. The graphics hardware system 108, over bus 100", supplies data and control signals to a local frame buffer memory 110 which refreshes the display device 105 for rendering images (including graphics images) on display device 105. *Id.* at col. 5, ll. 63-67.

Figure 2 is reproduced below.

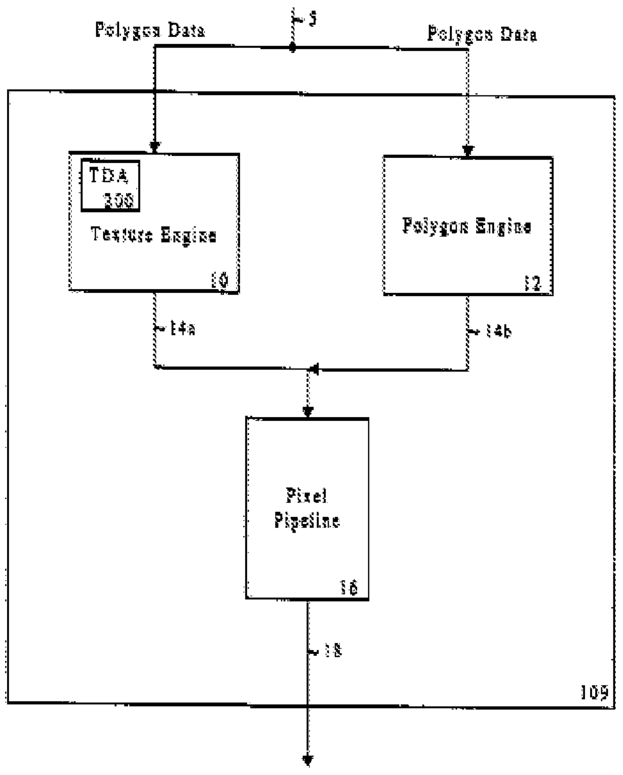

FIG. 2

Figure 2 is a block diagram of 3D graphics subunit 109. *Id.* at col. 2, ll. 42-43. As correctly noted by Appellants (Reply Br. 32), texture map data retrieval (TDA) circuit 200 in texture engine 10 is described as performing texture map *data* retrieval processes. *Id.* at col. 6, ll. 23-25.

The Examiner, citing Wang's column 5, lines 38-67, summarized above, found that graphics subunit 109 is a texture module that receives display instructions including data from local frame buffer 110, which corresponds to Rivard's DRAM 655:

> Wang . . . suggests a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; *based on the control signals send by the graphics hardware system, the frame buffer returns the polygon display instructions that includes [sic] the texture data, so that the graphics subunit executes the display instructions using this data*).

Final Action 6 (emphasis added). Appellants responded to this finding by arguing that:

> [t]he excerpt from Wang relied on by the Examiner merely discloses a "graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory." However, only generally disclosing that a graphics subunit executes instructions stored in computer memory, as in Wang, fails to specifically meet appellant's claimed "receiving instructions from the video memory in response to the instruction request <u>utilizing the texture module</u> . . ." (emphasis added), as appellant claims. In fact, appellant notes that Wang only discloses that the "graphics subunit 109 includ[es] a texture engine 10, [and] a polygon engine 12," and that the "texture engine 10 is responsible for retrieving the texture map data," whereas the "polygon engine 12 . . .

> performs well known polygon rendering functions" (Col. 6, lines 3-49). Thus, Wang clearly does not disclose "receiving instructions . . . <u>utilizing the texture module</u> . . ." (emphasis added), as claimed.

Br. 28-29 (brackets in original). We understand Appellants' position to be that Wang fails to disclose a texture module that receives instructions because the only part of graphics subunit 109 that can properly be characterized as a "texture module" is texture engine 10, which receives only data, with the result that the term "texture module" does not read on polygon engine, which performs well known polygon rendering functions (presumably by receiving and using instructions). However, Appellants have not established that it is unreasonable for the Examiner to characterize graphics subunit 109 as a whole as a "texture module," a term that is not defined in the claim or in Appellants' Specification.[4]

Turning now to the Examiner's proposed combination of reference teachings, we begin by noting that although the Examiner must show "'some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness,'" such reasoning "need not seek out precise

[4] The additional, new arguments regarding Wang that are presented at pages 32-33 of the Reply Brief will not be considered, because they were not necessitated by a new point in the Answer and thus should have been presented in the opening Brief. *See Optivus Technology, Inc. v. Ion Beam Applications S.A.*, 469 F.3d 978, 989 (Fed. Cir. 2006) (argument raised for the first time in the reply brief that could have been raised in the opening brief is waived); *accord*, *Ex parte Scholl*, No. 2007-3653, slip op. at 18 n.13 (BPAI March 13, 2008) (designated as an "Informative Opinion"),
(Continued on next page.)

teachings directed to the specific subject matter of the challenged claim." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007) (quoting *Kahn*, 441 F.3d at 988).

The Examiner stated the rationale for combining the teachings of Rivard and Wang as follows:

> [I]t would have been obvious to one of ordinary skill in art at the time of [the] present invention to have [Rivard's DRAM] memory [655] return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67).

Final Action 6-7. Appellants responded with a number of arguments, including that "Rivard does not recognize one of the various possible problems solved by appellant, namely to 'accommodate the programmability of recent texture and shader modules without being inhibited by the size of associated programs,' for example." Br. 14 (citing Specification 5:13-15). This argument is unconvincing because acceptable rationales for combining reference teachings are not limited to solving the problems the applicants were trying to solve. "[A]ny need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *In re ICON Health and Fitness, Inc.*, 496 F.3d 1374, 1380 (Fed. Cir. 2007) (quoting *KSR*, 550 U.S. at 420).

---

http://www.uspto.gov/web/ offices/dcom/bpai/its/fd073653.pdf.

Appellants (Br. 13) also argue that "[a]lthough a prior art device 'may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion or motivation in the reference to do so,'" citing *In re Mills*, 916 F.2d 680, 682 (Fed. Cir. 1990). However, "the [obviousness] analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ." *KSR*, 550 U.S. at 418.

Appellants also argue that storing instructions as well as data in Rivard's DRAM 655 "would result in no need for Rivard's purposefully included 'pipeline latency elements,' since the combination of data and instructions would arrive together" (Br. 13), and argue that such a modification of Rivard would be contrary to *In re Ratti*, 270 F.2d 810, 813 (CCPA 1959) (reversing rejection in part because "[t]his suggested combination of references would require a substantial reconstruction and redesign of the elements shown in Chinnery et al. as well as a change in the basic principles under which the Chinnery et al. construction was designed to operate") (*cited at* Manual of *Patent Examining Procedure* § 2143.01, subheading VI (8th ed., rev. 6, Sept. 2007)). The Examiner responded by explaining that it would have been obvious to retain Rivard's pipeline latency elements when Rivard is modified to store instructions as well as data in DRAM 655:

> [I]n the instance when the instructions and data are combined together, no pipeline latency element will be required;

> therefore, when the Rivard reference is modified as taught by Wang below, the modified system might have the data and the instructions coordinated when arrived at cache data store and memory data resolver, and will not require the use of pipeline latency element; however, *having the pipeline latency element will help to coordinate the instructions and data at their arrival at cache data store and memory data resolver, when there is a delay between the arrival of data and it's [sic] associated instructions*; it should be noted that Rivard has this pipeline latency element in addition to the claimed limitation, and moreover, the examiner interprets that the modified system with pipeline latency element will still function as effectively and generate the desired results, even when there is no delay between the arrival of the data and it's associated instructions).

Answer 12-13 (emphasis added). Although Appellants reproduced the above paragraph at pages 4-5 of the Reply Brief, the Reply Brief fails to address the reasoning stated therein, let alone demonstrate that it is erroneous. Instead, Appellants simply repeat their argument that eliminating the pipeline latency elements would be contrary to *Ratti*. *Id.* at 5-6.

We are also unpersuaded by Appellants' argument that "Rivard actually *teaches away* from applicant's claim language by intentionally incorporating the 'pipeline latency elements 1025' for the specific purpose of coordinating the arrival of the instructions and the memory data from <u>separate</u> sources." Br. 15-16. We do not agree that Rivard's disclosure of receiving the instructions and the memory data from different sources would have been understood to discourage a person from modifying Rivard so as to allow instructions and memory data to be obtained from a single source. *See ICON Health and Fitness*, 496 F.3d at 1381 ("A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant.") (citation omitted)).

Appellants further assert that:

> the mere disclosure of a memory request generator 1020 performing DRAM memory requests before the instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to suggest "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

Br. 26. This assertion is merely a conclusion that is unsupported by a detailed analysis. The same is true of the following assertion:

> Clearly, performing memory requests before instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to even suggest an "instruction request," much less "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

Id. at 26-27.

Regarding the requirement of claim 1 that the "texture module" send an "instruction request" to the video memory, Appellants, as noted above, argued that in Rivard without modification in view of Wang the "memory request" that texel cache system 650 sends to DRAM 655 is not an "instruction request." *See* Br. 19 ("[D]isclosing that a memory request is generated for all misses, as in Rivard, fails to teach 'sending an instruction request to the video memory, where a texture module . . . sends the instruction request to the video memory' (emphasis added), as claimed by appellant."). However, Appellants have not asserted that, let alone explained why, this claim limitation will not be satisfied when, as proposed by the Examiner, Rivard is modified so as to store instructions as well as texture data in DRAM 655.

Issue 1/Group 1 claims 1-12, 21, 24, and 27

For the foregoing reasons, Appellants have failed to show that the Examiner erred in concluding that the subject matter of representative claim 1 would have been obvious over Rivard in view of Wang. We are therefore affirming the rejection of that claim as well as the rejection of the other Issue 1/Group 1 claims (viz., claims 2-12, 21, 24, and 27).

Issue 1/Group 2 claims 25 and 26

Appellants only arguments (Br. 30-41) regarding the rejection of independent claims 25 and 26 are the above-discussed, unpersuasive arguments made with respect to claim 1. The rejection of claims 25 and 26 is affirmed.

Issue 1/Group 3 claim 28

Regarding independent claim 28, Appellants make only the same arguments they made against the rejection of claim 1. The rejection of claim 28 is affirmed.

Issue 1/Group 4 claim 30

Regarding independent claim 30, Appellants make only the same arguments they made against the rejection of claim 1. The rejection of claim 30 is affirmed.

Issue 1/Group 5 claim 18

Claim 18 reads:

> 18. The method as recited in claim 1, wherein a complete instruction set is received in response to the instruction request.

The Examiner concluded that the claim term "complete instruction set" is readable on a single instruction:

> The graphics subunit [109 in Wang] receives display instructions including texture data based on the supplied data and control signals (the examiner interprets that the claims do not specify how many instructions are needed to make a set complete; the examiner interprets that if the required operation can be performed from just the one received instruction, then it is considered as a complete set of instruction; the single display instruction received by the graphics from the sub-routine process in response to the control signals correspond to a complete set of instruction; this instruction in the sub-routine is executed by the graphics subunit for rendering the concerned

> graphics primitive, and therefore it is considered as a complete set of instruction). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return a complete set of instructions as taught by Wang to the texture module of Rivard because this instruction set is needed to render the concerned graphics primitive (col. 5 lines 43-45 and lines 63-67).

Final Action 11-12. Appellants do not appear to deny that the claim term "complete instruction set" is broad enough to read on a single instruction. Instead, they argue that "[c]learly, a <u>series</u> of display instructions [in Wang] fails to support 'just the <u>one received instruction</u>' (emphasis added), as alleged by the Examiner." Br. 64. This argument is unpersuasive because it is not responsive to the Examiner's position that it would have been obvious to receive and execute a series of display instructions one instruction at a time. The rejection of claim 18 is affirmed.

Issue 1/Group 6 claims 19 and 20

Claim 19 reads:

> 19. The method as recited in claim 1, wherein a partial instruction set is received in response to the instruction request.

The Examiner found that the term "partial instruction set" can be read on a single instruction of plurality of instructions:

> [T]he examiner interprets that if the required operation can be performed from just the one received instruction, then it is considered as a complete set of instruction; however, the rendering process might need other instructions for rendering

> other primitives, and in this case, the single received instruction is considered as a partial set of instruction[.]

Final Action 12. Appellants' arguments regarding claim 19 (Br. 64-66) do not address this position of the Examiner. The rejection of claim 19 is affirmed, as is the rejection of claim 20, which depends on claim 19 and is not separately argued.

Issue 2/Group 1 claims 13-17, 22, and 23

Appellants argue only that dependent claims 13-17, 22, and 23 are patentable over the applied prior art for the same reasons as claim 1 (Br. 67). The rejection of those claims is therefore affirmed. *In re Nielson*, 816 F.2d 1567, 1572 (Fed. Cir. 1987).

Issue 2/Group 2 claim 29

Regarding independent claim 29, Appellants (Br. 67-78) make only the same arguments they made against the rejection of claim 1. The rejection of claim 29 is therefore affirmed.

DECISION

The Examiner's rejections of claims 1-30 under 35 U.S.C. § 103(a) for obviousness over the applied prior art are affirmed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136. *See* 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED

KIS

Zilka-Kotab, PC
P.O. BOX 721120
SAN JOSE, CA 95172-1120